WORK WITH AN EXPERT SYSTEM

Allison Hahn
Honors Program Capstone
Spring 1985

For my Capstone, I assisted Computer Science Instructor Kathi Davis with her implementation of an expert system to translate a conventional file system into a commercial database system. The main purpose of her expert system is to give users of conventional file systems a tool to use in the conversion to commercial database systems; an easy tool that will make the process less painful than it is today. The conventional file system is first translated into a conceptual Entity-Relationship model and this conceptual model is then translated into a commercial database system. The Entity-Relationship conceptual model assists the user in understanding what currently exists within his environment and helps in forming a new conceptual model of the desired database environment. (1)

The first project I worked on dealt with the timing function in Prolog. Prolog is a programming language. The word Prolog stands for programming logic. Logic programming is a movement toward use of a form of clausal logic as a high-level programming language. Prolog is perhaps the best known of such languages and comes closest to the logic programming ideal. (2) The most fundamental principle of logic programming is that a program is a set of logic propositions explaining the relationship between data. The propositions are expressed in Horn Clauses. The simplest clause is the assertion of a fact which consists of a conclusion followed by no conditions. Here is an example of a fact: male(Henry). This can be read as 'Henry is male'. Clauses can also be rules. Rules show the dependence of one fact upon another. For example: male(Henry) if father(Henry). This rule can be read as 'If Henry is a father, Henry is male'. This

rule implies that if it is known that Henry is a father then the knowledge that Henry is male can be proven. Prolog is a good language for expressing relational database theory and implementing expert systems.

In Prolog, the timing function is used to tell the user how long a routine or parts of routine of a Prolog program take to execute. Kathi had a program written in Prolog that parsed COBOL file layouts. The program's run time was close to twelve hours which was too long to be feasible. By inserting the timing function in several key routines, I was able to determine the area causing the program to take such a long time to execute. Exhibit 1 shows the format of the Timing Function. The PP clause instructs Prolog to pring out the hours, minutes, and seconds when the timing function is executed. Exhibit 2 shows the output from a part of the parsing program which was executed with the timing functions inserted.

My next project involved using a graphics package to graph a pre-designed Entity-Relationship model. I was given a rough sketch of the layout of the model and I used BASIC to write a program to produce the graph. The graphics package, called Halo Graphics, had commands built into it to produce various geometric figures, draw lines, and move the cursor to certain points on the screen. It was not a very powerful graphics package in the sense that I had to specify specific coordinate locations on the screen before issuing a command. This meant that I had to use a grid, and map out exactly where each point of the graph had to be before even starting to write the BASIC program to implement the

graph. Exhibit 3 shows the BASIC program I wrote to produce the Entity-Relationship model. Exhibit 4 shows the Entity-Relationship model the BASIC program produced.

The majority of my time was spent writing a Prolog program to traverse a tree structure. The goal of my program was to take a tree structure which described a conventional file system and reduce it to only its necessary components. The tree structure was drawn from Prolog rules describing relationships among the pieces of data in the file. Whenever I wanted to alter the tree structure in any way, I had to alter the appropriate Prolog rules. When I was finished altering the rules, I had to be able to look at the new set of rules and produce the reduced tree structure. We decided to traverse the tree in pre order. A pre order traversal algorithm for a tree structure is a algorithm in which a node is process before both of its children. For example, after the root node is processed, the left subtree is processed and then the right is processed. Kathi also gave me a list of exceptions to follow for when a part of the tree structure should not be combined. Exhibit 5 shows the tree structure before processing and Exhibit 6 shows its accompanying Prolog rules. Exhibit 7 shows what the new tree looks like. Finally, Exhibit 8 shows the actual Prolog code I wrote to alter the tree structure.

While it was not my intention to explain in great detail each of the projects I completed while working with Kathi Davis, I hope that I have given you an overview of the work I have done this semester for my Capstone. I was very glad to have been given the opportunity to do an independent study in computer

science. It was an experience which I am not so sure I would have pursued on my own had it not been for the Honors Program. Now I feel that I have a good grasp of programming in Prolog since I have had actual hands-on experience. I think that I am very lucky to have had the opportunity to learn Prolog because I believe that it has the potential to become the most widely used language in the future. I think that it has such potential not only because of Japan's heavy use of Prolog, but because Prolog is the language most often used in the United States to implement expert systems.

While I was searching for the subject of my Capstone, I came across one or two other possibilities in which future computer science honors students my be interested. The content of the following projects may not be obvious to you, but if you mention them to a computer science students they will most likely understand what the projects involve.

1. Revies the old 465 Examples. (They about six years old and very out of date.)
2. Put together a book of example programs to be used in 466. (IMS and CICS examples.)

Not only would these projects be interesting to work on, but they would also improve the learning material presently being used in computer science classes which in turn would directly benefit other computer science students.

FOOTNOTES

[1] Kathi Hogshead Davis, "An Experimental Expert System to Translate a Conventional File System into a Commercial Database System" (Master's Thesis, Illinois Institute of Technology, 1985), p. 1.

[2] "Logic Programming Through Prolog, Steps into the Real World," Systems & Software, Jan 1985, pp. 147-148.

BIBLIOGRAPHY

Davis, Kathi Hogshead. "An Experimental Expert System to Translate a Conventional File System into a Commercial Database System," Master's Thesis, Illinois Institute of Technology, 1985.

"Logic Programming Through Prolog, Steps into the Real World." Systems & Software, Jan 1985, pp. 147-151.

```
A>dir b:
SECREC2            4352    1-09-85   12:02a
SECREC   EMU      54016    1-04-85   12:06a
AUGFILE  IN        4352    1-09-85   12:02a
CPMDRIVR LOG      19200   12-18-84    1:44a
SPACEOUT TST      19712    1-14-85   12:13a
       5 File(s)
list timer-list
timer-list if
     TIME (X Y Z) and
     (hour X) PP and
     (min Y) PP and
     (sec Z) PP
A.
```

EXHIBIT 1

```
(which(s:test-driver)
(hour 0)
(min 22)
(sec 25)
(hour 0)
(min 22)
(sec 37)
(hour 0)
(min 22)
(sec 42)
(hour 0)
(min 24)
(sec 23)
((token 1 "01"))
((token 2 FIRST-RECORD))
((token 3 .))
(hour 0)
(min 24)
(sec 26)
(hour 0)
(min 24)
(sec 55)
((workname FIRST-RECORD))
((workform 0))
((worklen (0 0)))
((worklev "01"))
((worklslev 0))
((workocc N))
((workredef N))
((worksister none))
(hour 0)
(min 25)
(sec 6)
(hour 0)
(min 25)
(sec 35)
(hour 0)
(min 25)
(sec 40)
(hour 0)
(min 28)
(sec 10)
((token 1 "02"))
((token 2 NUMERIC-FIELD-1))
((token 3 PIC))
((token 4 "9(7)"))
((token 5 .))
(hour 0)
(min 28)
(sec 14)
(hour 0)
(min 29)
(sec 5)
((workname NUMERIC-FIELD-1))
((workform N))
((worklen (7 0)))
((worklev "02"))
((worklslev "01"))
((workocc N))
((workredef N))
((worksister none))
(hour 0)
(min 29)
(sec 15)
```

EXHIBIT 2

```
100 '-------------------------------------------------------------------------
110 '  Haloi.bas          program statements necessary to use HALO in
115 '                     Basic interpreter
120 '-------------------------------------------------------------------------
130 DEFINT A-M                                  ' (Any deftype statements should go
140                                             ' here.  Your choice.)
160 GOSUB 8000                                  ' need to be changed depending on
170 DEF SEG = &H2300
180 BLOAD "haloi.bin",0                         ' Basic manual for details.
190 '-------------------------------------------------------------------------
200 '  Your basic program should follow this comment and precede line 8000.
220 '  The statements beginning at line 8000 are necessary.
230 '  They define the offsets to the Halo routines.
240 '-------------------------------------------------------------------------
250 MODE% = 1
260 CALL INITGRAPHICS(MODE%)
270 INTZERO% = 0
280 INTONE% = 1
290 CALL SETTEXTCLR (INTONE%,INTZERO%)
300 INTFOR% = 4
310 INTTWO% = 2
320 CALL INITTCUR (INTFOR%,INTTWO%,INTONE%)
330 CALL SETTEXT (INTONE%,INTONE%,INTZERO%,INTZERO%)
340 X1% = 0 : Y1% = 34
350 X2% = 56 : Y2% = 50
360 GOSUB 2000
370 X1% = 0 : Y1% = 109
380 X2% = 96 : Y2% = 125
390 GOSUB 2000
400 X1% = 350 : Y1% = 34
410 X2% = 446 : Y2% = 50
420 GOSUB 2000
430 X1% = 350 : Y1% = 109
440 X2% = 486 : Y2% = 125
450 GOSUB 2000
460 X1% = 250 : Y1% = 183
470 X2% = 402 : Y2% = 199
480 GOSUB 2000
490 X1% = 450 : Y1% = 183
500 X2% = 562 : Y2% = 199
510 GOSUB 2000
520 XTCUR% = 4 : YTCUR% = 46
530 TEXT$ = "people"
540 GOSUB 5000
550 XTCUR% = 4 : YTCUR% = 121
560 TEXT$ = "major-table"
570 GOSUB 5000
580 XTCUR% = 354 : YTCUR% = 46
590 TEXT$ = "crse-record"
600 GOSUB 5000
610 XTCUR% = 354 : YTCUR% = 121
620 TEXT$ = "crse-record-time"
630 GOSUB 5000
640 XTCUR% = 254 : YTCUR% = 195
650 TEXT$ = "crse-meet-specific"
660 GOSUB 5000
670 XTCUR% = 454 : YTCUR% = 195
680 TEXT$ = "crse-meetings"
690 GOSUB 5000
700 SXCUR% = 28 : SYCUR% = 50
705 X1CUR% = 28 : Y1CUR% = 70
710 X2CUR% = 52 : Y2CUR% = 80
```

EXHIBIT 3

```
730 X4CUR% = 4  : Y4CUR% = 80
735 X5CUR% = 28 : Y5CUR% = 70
740 FXCUR% = 28 : FYCUR% = 109
745 GOSUB 6000
750 SXCUR% = 398 : SYCUR% = 50
755 X1CUR% = 398 : Y1CUR% = 70
760 X2CUR% = 422 : Y2CUR% = 80
765 X3CUR% = 398 : Y3CUR% = 90
770 X4CUR% = 374 : Y4CUR% = 80
775 X5CUR% = 398 : Y5CUR% = 70
780 FXCUR% = 398 : FYCUR% = 109
785 GOSUB 6000
790 SXCUR% = 395 : SYCUR% = 125
795 X1CUR% = 326 : Y1CUR% = 145
800 X2CUR% = 350 : Y2CUR% = 155
805 X3CUR% = 326 : Y3CUR% = 165
810 X4CUR% = 302 : Y4CUR% = 155
815 X5CUR% = 326 : Y5CUR% = 145
820 FXCUR% = 326 : FYCUR% = 183
825 GOSUB 6000
830 SXCUR% = 440 : SYCUR% = 125
835 X1CUR% = 506 : Y1CUR% = 145
840 X2CUR% = 530 : Y2CUR% = 155
845 X3CUR% = 506 : Y3CUR% = 165
850 X4CUR% = 482 : Y4CUR% = 155
855 X5CUR% = 506 : Y5CUR% = 145
860 FXCUR% = 506 : FYCUR% = 183
865 GOSUB 6000
870 SXCUR% = 56 : SYCUR% = 34
875 X1CUR% = 176 : Y1CUR% = 10
880 X2CUR% = 200 : Y2CUR% = 0
885 X3CUR% = 224 : Y3CUR% = 10
890 X4CUR% = 200 : Y4CUR% = 20
895 X5CUR% = 176 : Y5CUR% = 10
900 FXCUR% = 350 : FYCUR% = 34
905 GOSUB 6000
910 SXCUR% = 56 : SYCUR% = 50
920 X1CUR% = 176 : Y1CUR% = 74
925 X2CUR% = 200 : Y2CUR% = 64
930 X3CUR% = 224 : Y3CUR% = 74
935 X4CUR% = 200 : Y4CUR% = 84
940 X5CUR% = 176 : Y5CUR% = 74
945 FXCUR% = 350 : FYCUR% = 50
950 GOSUB 6000
951 X1% = 28 : Y1% = 55
952 X2% = 38 : Y2% = 50
953 X3% = 18 : Y3% = 50
954 XC1% = 26 : YC1% = 52
955 XC2% = 30 : YC2% = 52
956 GOSUB 4000
957 X1% = 398 : Y1% = 104
958 X2% = 408 : Y2% = 109
959 X3% = 388 : Y3% = 109
960 XC1% = 400 : YC1% = 107
961 XC2% = 396 : YC2% = 107
962 GOSUB 4000
1000 SXCUR% = 402 : SYCUR% = 191
1010 FXCUR% = 450 : FYCUR% = 191
1020 GOSUB 3000
1030 X1% = 70 : Y1% = 31
1040 X2% = 56 : Y2% = 40
1050 X3% = 40 : Y3% = 34
1060 XC1% = 57 : YC1% = 35
1070 XC2% = 55 : YC2% = 33
1080 GOSUB 4000
```

```
1100 X2% = 56 : Y2% = 44
1110 X3% = 40 : Y3% = 50
1120 XC1% = 57 : YC1% = 49
1130 XC2% = 57 : YC2% = 51
1140 GOSUB 4000
1150 X1% = 336 : Y1% = 31
1160 X2% = 350 : Y2% = 40
1170 X3% = 366 : Y3% = 34
1180 XC1% = 349 : YC1% = 35
1190 XC2% = 349 : YC2% = 33
1200 GOSUB 4000
1210 X1% = 336 : Y1% = 53
1220 X2% = 350 : Y2% = 44
1230 X3% = 366 : Y3% = 50
1240 XC1% = 349 : YC1% = 49
1250 XC2% = 349 : YC2% = 51
1260 GOSUB 4000
1700  X1% = 200 : Y1% = 600
1710 GOSUB 6200
1800 DEV% = 1
1810 CALL SETGPRINT (DEV%)
1820 CALL GPRINT
1998 CALL CLOSEGRAPHICS
1999 END
2000 CALL MOVABS (X1%,Y1%)
2010 CALL BOX (X1%,Y1%,X2%,Y2%)
2020 RETURN
3000 CALL MOVABS (SXCUR%,SYCUR%)
3010 CALL LNABS (FXCUR%,FYCUR%)
3020 RETURN
4000 CALL MOVABS (X1%,Y1%)
4010 CALL LNABS (X2%,Y2%)
4020 CALL MOVABS (X1%,Y1%)
4030 CALL LNABS (X3%,Y3%)
4040 CALL MOVABS (XC1%,YC1%)
4050 CALL FILL (INTONE%)
4060 CALL MOVABS (XC2%,YC2%)
4070 CALL FILL (INTONE%)
4080 RETURN
5000 CALL MOVTCURABS (XTCUR%,YTCUR%)
5010 CALL TEXT (TEXT$)
5020 RETURN
6000 CALL MOVABS (SXCUR%,SYCUR%)
6010 CALL LNABS (X1CUR%,Y1CUR%)
6020 CALL LNABS (X2CUR%,Y2CUR%)
6030 CALL LNABS (X3CUR%,Y3CUR%)
6040 CALL LNABS (X4CUR%,Y4CUR%)
6050 CALL LNABS (X5CUR%,Y5CUR%)
6060 CALL MOVABS (X3CUR%,Y3CUR%)
6070 CALL LNABS (FXCUR%,FYCUR%)
6080 RETURN
6200 CALL MOVTCURABS (X1%,Y1%)
6210 RETURN
8000 ARC                                    = &H557F
8001 BAR                                    = &H3863
8002 BOX                                    = &H3866
8003 CIR                                    = &H5579
8004 CLOSEG                                 = &H2EA
8005 CLOSEGRA                               = &H2EA
8006 CLOSEGRAPHICS                          = &H2EA
8007 CLR                                    = &H2470
```

major-table
people
crse-meet-specific
crse-meetings
crse-record-time
crse-record

EXHIBIT 4

PEOPLE
PEOPLE - INSTR - DATA
PEOPLE - STUD - DATA
COURSE - TEACHERS
MAJOR - TRACK

EXHIBIT 5

```
list all
pdutab ((PEOPLE) 0 1 none 1)
pdutab ((PEOPLE-INSTR-DATA) (PEOPLE) 2 none 2)
pdutab ((PEOPLE-STUDENT-DATA) (PEOPLE) 2 none 2)
pdutab ((PEOPLE-STUDENT-GRADUATION) (PEOPLE-STUDENT-DATA) 2 none 2)
pdutab ((PEOPLE-STUDENT-CRSES) (PEOPLE-STUDENT-DATA) 3 none 2)
pdutab ((CRSE-RECORD) 0 6 none 1)
pdutab ((CRSE-STUDENTS) (CRSE-RECORD) 3 none 2)
pdutab ((CRSE-TEACHERS) (CRSE-RECORD) 3 none 2)
pdutab ((CRSE-RECORD-TIMES) (CRSE-RECORD) 7 none 2)
pdutab ((CRSE-MEETINGS) (CRSE-RECORD-TIMES) 4 (CRSE-MEET-SPECIFIC) (2 3))
pdutab ((CRSE-MEET-SPECIFIC) (CRSE-RECORD-TIMES) 4 (CRSE-MEETINGS) (2 3))
pdutab ((MAJOR-TABLE) 0 1 none 1)
pduref ((PEOPLE) (PEOPLE-ID) (PEOPLE-INSTR-DATA) (PEOPLE-ID) (4 4) (1 -> 1))
pduref ((PEOPLE) (PEOPLE-ID) (PEOPLE-STUDENT-DATA) (PEOPLE-ID) (4 4) (1 -> 1))
pduref ((PEOPLE-STUDENT-DATA) (PEOPLE-ID) (PEOPLE-STUDENT-GRADUATION) (PEOPLE-ID
) (4 4) (1 -> 1))
pduref ((PEOPLE-STUDENT-GRADUATION) (PEOPLE-STUD-GRAD-MAJOR-CODE) (MAJOR-TABLE)
(MAJOR-CODE) (1 0) (M -> 1))
pduref ((PEOPLE-STUDENT-DATA) (PEOPLE-ID) (PEOPLE-STUDENT-CRSES) (PEOPLE-ID) (4
4) (1 -> m))
pduref ((PEOPLE-STUDENT-CRSES) (PEOPLE-STUD-CRSES-ID) (CRSE-RECORD) (CRSE-ID) (5
 0) (M -> 1))
pduref ((CRSE-RECORD) (CRSE-ID) (CRSE-STUDENTS) (CRSE-ID) (4 4) (1 -> m))
pduref ((CRSE-STUDENTS) (CRSE-STUDENTS) (PEOPLE) (PEOPLE-ID) (1 0) (M -> 1))
pduref ((CRSE-RECORD) (CRSE-ID) (CRSE-TEACHERS) (CRSE-ID) (4 4) (1 -> m))
pduref ((CRSE-TEACHERS) (CRSE-TEACHERS) (PEOPLE) (PEOPLE-ID) (1 0) (M -> 1))
pduref ((CRSE-RECORD) (CRSE-ID) (CRSE-RECORD-TIMES) (CRSE-ID) (5 0) (1 -> m))
pduref ((CRSE-RECORD-TIMES) (CRSE-ID CRSE-TIMES-ID) (CRSE-MEETINGS) (CRSE-ID CRS
E-TIMES-ID) (4 4) (1 -> 1))
pduref ((CRSE-MEETINGS) (CRSE-ID CRSE-TIMES-ID) (CRSE-MEET-SPECIFIC) (CRSE-ID CR
SE-TIMES-ID) (0 0) (0 -> 0))
pduref ((CRSE-RECORD-TIMES) (CRSE-ID CRSE-TIMES-ID) (CRSE-MEET-SPECIFIC) (CRSE-I
D CRSE-TIMES-ID) (4 4) (1 -> 1))
pduref ((CRSE-MEET-SPECIFIC) (CRSE-ID CRSE-TIMES-ID) (CRSE-MEETINGS) (CRSE-ID CR
SE-TIMES-ID) (0 0) (0 -> 0))
pdufields-of ((PEOPLE-ID A (9 0) 02) (PEOPLE))
pdufields-of ((PEOPLE-NAME A (20 0) 02) (PEOPLE))
pdufields-of ((PEOPLE-INSTR-TITLE A (2 0) 03) (PEOPLE-INSTR-DATA))
pdufields-of ((PEOPLE-INSTR-DEPT A (4 0) 03) (PEOPLE-INSTR-DATA))
pdufields-of ((PEOPLE-STUDENT-CLASS A (2 0) 03) (PEOPLE-STUDENT-DATA))
pdufields-of ((PEOPLE-STUD-GRAD-MAJOR-CODE A (4 0) 04) (PEOPLE-STUDENT-GRADUATIO
N))
pdufields-of ((PEOPLE-STUD-GRAD-DATE A (8 0) 04) (PEOPLE-STUDENT-GRADUATION))
pdufields-of ((PEOPLE-STUD-CRSES-ID A (6 0) 04) (PEOPLE-STUDENT-CRSES))
pdufields-of ((PEOPLE-STUD-CRSES-CREDIT N (3 0) 04) (PEOPLE-STUDENT-CRSES))
pdufields-of ((PEOPLE-STUD-CRSES-GRADE A (1 0) 04) (PEOPLE-STUDENT-CRSES))
pdufields-of ((CRSE-RECORD-TYPE A (1 0) 02) (CRSE-RECORD))
pdufields-of ((CRSE-ID A (6 0) 02) (CRSE-RECORD))
pdufields-of ((CRSE-TITLE A (20 0) 02) (CRSE-RECORD))
pdufields-of ((CRSE-STUDENTS A (9 0) 02) (CRSE-STUDENTS))
pdufields-of ((CRSE-TEACHERS A (9 0) 02) (CRSE-TEACHERS))
pdufields-of ((CRSE-TIMES-RECORD-TYPE A (1 0) 02) (CRSE-RECORD-TIMES))
pdufields-of ((CRSE-TIMES-ID A (6 0) 02) (CRSE-RECORD-TIMES))
pdufields-of ((CRSE-MEETINGS A (20 0) 02) (CRSE-MEETINGS))
pdufields-of ((CRSE-MEETING-TIMES A (12 0) 03) (CRSE-MEET-SPECIFIC))
pdufields-of ((CRSE-MEETING-PLACE A (8 0) 03) (CRSE-MEET-SPECIFIC))
pdufields-of ((MAJOR-CODE A (4 0) 02) (MAJOR-TABLE))
pdufields-of ((MAJOR-TITLE A (30 0) 02) (MAJOR-TABLE))
pdufields-of ((PEOPLE-ID A (9 0) 0) (PEOPLE-INSTR-DATA))
pdufields-of ((PEOPLE-ID A (9 0) 0) (PEOPLE-STUDENT-DATA))
pdufields-of ((PEOPLE-ID A (9 0) 0) (PEOPLE-STUDENT-GRADUATION))
pdufields-of ((PEOPLE-ID A (9 0) 0) (PEOPLE-STUDENT-CRSES))
pdufields-of ((CRSE-ID A (6 0) 0) (CRSE-STUDENTS))
```

EXHIBIT 6

```
pdufields-of ((CRSE-ID A (6 0) 0) (CRSE-TEACHERS))
pdufields-of ((CRSE-ID A (6 0) 0) (CRSE-RECORD-TIMES))
pdufields-of ((CRSE-ID A (6 0) 0) (CRSE-MEETINGS))
pdufields-of ((CRSE-TIMES-ID A (6 0) 0) (CRSE-MEETINGS))
pdufields-of ((CRSE-ID A (6 0) 0) (CRSE-MEET-SPECIFIC))
pdufields-of ((CRSE-TIMES-ID A (6 0) 0) (CRSE-MEET-SPECIFIC))
pdukey ((PEOPLE) (PEOPLE-ID) 2)
pdukey ((PEOPLE-INSTR-DATA) (PEOPLE-ID) 4)
pdukey ((PEOPLE-STUDENT-DATA) (PEOPLE-ID) 4)
pdukey ((PEOPLE-STUDENT-GRADUATION) (PEOPLE-ID) 4)
pdukey ((PEOPLE-STUDENT-CRSES) (PEOPLE-ID PEOPLE-STUD-CRSES-ID) 4)
pdukey ((CRSE-RECORD) (CRSE-ID) 3)
pdukey ((CRSE-STUDENTS) (CRSE-ID CRSE-STUDENTS) 4)
pdukey ((CRSE-TEACHERS) (CRSE-ID CRSE-TEACHERS) 4)
pdukey ((CRSE-RECORD-TIMES) (CRSE-ID CRSE-TIMES-ID) 4)
pdukey ((CRSE-MEETINGS) (CRSE-ID CRSE-TIMES-ID) 4)
pdukey ((CRSE-MEET-SPECIFIC) (CRSE-ID CRSE-TIMES-ID) 4)
pdukey ((MAJOR-TABLE) (MAJOR-CODE) 2)
protocol-of (6 ((PEOPLE-STUDENT-GRADUATION)))
protocol-of (9 ((MAJOR-TABLE)))
protocol-of (6 ((PEOPLE-STUDENT-CRSES)))
protocol-of (8 ((CRSE-RECORD)))
protocol-of (6 ((CRSE-STUDENTS)))
protocol-of (8 ((PEOPLE)))
protocol-of (6 ((CRSE-TEACHERS)))
protocol-of (8 ((PEOPLE)))
in-file ((PEOPLE) STUDENTINFO)
in-file ((PEOPLE-INSTR-DATA) STUDENTINFO)
in-file ((PEOPLE-STUDENT-DATA) STUDENTINFO)
in-file ((PEOPLE-STUDENT-GRADUATION) STUDENTINFO)
in-file ((PEOPLE-STUDENT-CRSES) STUDENTINFO)
in-file ((CRSE-RECORD) CRSEFILE)
in-file ((CRSE-STUDENTS) CRSEFILE)
in-file ((CRSE-TEACHERS) CRSEFILE)
in-file ((CRSE-RECORD-TIMES) CRSEFILE)
in-file ((CRSE-MEETINGS) CRSEFILE)
in-file ((CRSE-MEET-SPECIFIC) CRSEFILE)
in-file ((MAJOR-TABLE) MAJORFILE)
combine if
     pdutab (PDUNAME X Y Z x) and
     ^PDUNAME testchild and
```

EXHIBIT 7

```
list all
combine if
      (9 map-flag) add and
      / and
      pdutab (PDUNAME GP X Y Z) and
      (pdu PDUNAME) PP and
      (gp GP) PP and
      PDUNAME proc-current-parent GP and
      FAIL and
      (PDUNAME GP) vars
PDUNAME proc-current-parent GP if
      PDUNAME proc-ck-m-1 and
      O map-flag and
      pdutab (CHILD PDUNAME X Y Z) and
      proc-consider-combining (PDUNAME CHILD GP) and
      / and
      (PDUNAME GP CHILD MAPPING) vars
proc-consider-combining (PDUNAME CHILD GP) if
      pdutab (CHILD X Y none Z) and
      process-see-if-children (CHILD PDUNAME GP) and
      / and
      (PDUNAME CHILD GP) vars
process-see-if-children (CHILD PDUNAME GP) if
      not pdutab (X CHILD Y Z x) and
      process-single (PDUNAME CHILD GP) and
      / and
      (PDUNAME CHILD GP) vars
process-see-if-children (CHILD PDUNAME GP) if
      CHILD proc-current-parent PDUNAME and
      / and
      (PDUNAME GP CHILD) vars
No sentences for proc-see-if-children
process-single (PDUNAME CHILD GP) if
      pdutab (PDUNAME O X Y Z) and
      (root1 PDUNAME O) PP and
      (pdutab (CHILD PDUNAME x y z)) delete and
      (pduref (PDUNAME X1 CHILD Y1 Z1 x1)) delete and
      not proc-pdufields-root-loop (PDUNAME CHILD) and
      / and
      (PDUNAME CHILD GP) vars
process-single (PDUNAME CHILD GP) if
      not proc-children-loop (PDUNAME GP) and
      / and
      (PDUNAME GP CHILD) vars
proc-pdufields-root-loop (PDUNAME CHILD) if
      pdufields-of (FIELD CHILD) and
      (root-2-child CHILD) PP and
      proc-pdufields-root (PDUNAME CHILD FIELD) and
      FAIL and
      (PDUNAME CHILD FIELD) vars
proc-pdufields-root (PDUNAME CHILD FIELD) if
      (root-3-child CHILD) PP and
      (pdufields-of-new (FIELD PDUNAME)) add and
      (root-3-pduname PDUNAME) PP and
      / and
      (PDUNAME CHILD FIELD) vars
No sentences for process-pdufields-replace
proc-pdufields-replace (GP PDUNAME CHILD FIELD) if
      (single3-child CHILD) PP and
      (pdufields-of-new (FIELD GP)) add and
      (single3-gp GP) PP and
      / and
```

EXHIBIT 8

```
(PDUNAME GP CHILD FIELD) vars
No sentences for process-single-temp
No sentences for proc-pdufields-replace-temp
proc-pdufields-replace-loop (GP PDUNAME CHILD) if
      pdufields-of (FIELD CHILD) and
      (single2-child CHILD) PP and
      proc-pdufields-replace (GP PDUNAME CHILD FIELD) and
      FAIL and
      (PDUNAME GP CHILD FIELD) vars
PDUNAME proc-ck-m-1 if
      pduref (PDUNAME X Y Z x (M -> 1)) and
      (y map-flag) delete and
      (1 map-flag) add and
      (map-flag-m-1-found 1) PP and
      / and
      (PDUNAME) vars
PDUNAME proc-ck-m-1 if
      (X map-flag) delete and
      (O map-flag) add and
      (map-flag-m-1-notfound O) PP and
      / and
      (PDUNAME) vars
proc-children-all (PDUNAME GP CHILD TYPE SISTER UPM) if
      (pdutab-new (CHILD GP TYPE SISTER UPM)) add and
      pduref (PDUNAME X CHILD Y Z x) and
      (pduref-new (GP X CHILD Y Z x)) add and
      (single1-gp-pduname-child GP PDUNAME CHILD) PP and
      not proc-pdufields-replace-loop (GP PDUNAME CHILD) and
      / and
      (PDUNAME GP CHILD TYPE SISTER UPM) vars
proc-children-loop (PDUNAME GP) if
      pdutab (CHILD PDUNAME TYPE SISTER UPM) and
      proc-children-all (PDUNAME GP CHILD TYPE SISTER UPM) and
      FAIL and
      (PDUNAME GP CHILD TYPE SISTER UPM) vars
No sentences for process-single-1
&.
```